View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by OJS Universitas Mulia (E-Journals)

OPTIMASI QUERY SEDERHANA GUNA KECEPATAN QUERY PADA DATABASE SERVER

Suhartati, Yeyen Dwi Atma
Magister Teknik Informatika STMIK AMIKOM Yogyakarta
e-mail : suhar_taty@yahoo.com, yeyenduwy@gmail.com

ABSTRAK

Informasi sebagai keluaran sebuah aplikasi akan sangat berguna jika memenuhi tiga aspek dalam kualitas yaitu relevan, akurat dan tepat waktu. Untuk mencapai kualitas dan menyesuaikan dengan tuntutan jaman yang serba cepat maka diperlukan optimasi yang baik dalam menampilkan data kembali. Pernyataan SQL dapat digunakan untuk mengambil data dari database manapun. Untuk mendapatkan hasil yang sama kita perlu menulis query SQL yang berbeda. Untuk kinerja yang lebih baik, perlu menggunakan kueri terbaik, cepat, dan efisien. Jadi kita perlu query SQL tuning berdasarkan kebutuhan bisnis dan pengguna. Makalah ini membahas bagaimana query SQL dapat dioptimalkan untuk kinerja yang lebih baik. Dalam tulisan ini tidak fokus mendalam tentang database tapi penggunaan dan trik query sederhana yang bisa diterapkan untuk mendapatkan keuntungan kinerja secara optimal.

Kata kunci : SQL, informasi, optimasi, query

1. PENDAHULUAN

Dalam sebuah aplikasi, semakin besar data yang tersimpan pada database maka dibutuhkan kecepatan akses yang memadai untuk mengelola informasi didalamnya. Disamping upgrade hardware yang cukup memakan biaya untuk menambah kecepatan proses, sebagai DBA kita juga perlu memahami apakah sudah evektif serta evisien pengelolaan data yang seharusnya ditampilkan oleh aplikasi.

Dalam pengelolaan data yang akan disajikan sebuah aplikasi, pernyataan SQL dapat digunakan untuk mengambil data dari database. Untuk mendapatkan hasil yang sama kita perlu menulis query SQL yang berbeda dari sebuah query yang mungkin masih standar.

Cara terbaiknya untuk menyempurnakan kinerja adalah dengan mencoba menuliskan sintak dengan berbagai cara dan membandingkan kecepatan pembacaan dan eksekusi yang dijalankan.

Pengoptimalan query merupakan keahlian penting bagi pengembang SQL dan administrator basis data (DBA). Untuk meningkatkan kinerja query SQL, pengembang dan DBA perlu memahami optimasi query dan teknik yang digunakannya untuk memilih jalur akses dan menyiapkan rencana eksekusi query.

Penyesuaian query melibatkan pengetahuan tentang teknik seperti pengoptimalan berbasis biaya dan pengoptimalan berbasis heuristik, ditambah alat yang disediakan platform SQL untuk menjelaskan rencana eksekusi kueri.

Berdasarkan latar belakang permasalahan tersebut, maka penulis tertarik untuk bagaimana memberikan masukan mengenai implementasi query pada database, yang bisa dimanfaatkan untuk kecepatan aplikasi.

2. LITERATUR

DATABASE

Basis data (database) merupakan kumpulan data yang saling berelasi. Data sendiri merupakan fakta mengenai obyek, orang, dan lain-lain. Data dinyatakan dengan nilai (angka, deretan karakter, atau symbol). Tujuan database adalah untuk mengatur data sehingga diperoleh kemudahan, ketepatan, dan kecepatan dalam pengambilan kembali [4].

SQL

SQL merupakan singkatan dari *Structured Query Language*. Pada dasarnya SQL merupakan bahasa komputer standar yang diterapkan oleh ANSI (American National Standard Institud) untuk mengakses dan memanipulasi sistem database.

SQL mempunyai dua macam perintah yang digunakan untuk mengelola dan mengorganisasikan basis data, yaitu :

1. Data Definition Language **(DDL)**

DDL atau *Data Definition Language* merupakan perintah SQL yang digunakan untuk mendefinisikan atau mendeklaraikan objek database, menciptakan objek database atau bahkan menghapus objek database. Objek database dapat berupa table atau database itu sendiri. DDL juga dapat digunakan untuk membuat koneksi antar table database beserta batasannya dengan menentukan indeks sebagai kuncinya. DDL yang umum digunakan adalah CREATE, DROP,ALTER [2].

2. Data Manipulation Language **(DML)**

DML atau Data Manipulation Language merupakan query yang digunakan untuk memanipulasi data, seperti untuk menampilkan data, mengubah data, atau mengisi data.

DML pada dasarnya dibagi menjadi dua :

a. Prosedural, yang menuntut pengguna menentukan data apa saja yang diperlukan dan bagaimana cara mendapatkannya.

b. Nonprosedural, yang menuntut pengguna menentukan data apa saja yang diperlukan, tetapi tidak perlu menyebutkan cara mendapatkannya [1].

Dalam melakukan optimasi, desain aplikasi saja tidak cukup untuk meningkatkan kerja, harus didukung dengan optimasi dari perintah SQL yang digunakan pada aplikasi tersebut. Dalam mendesain database, seringkali lokasi fisik data tidak menjadi perhatian penting. Karena hanya desain logik saja yang diperhatikan. Padahal untuk menampilkan hasil query dibutuhkan pencarian yang melibatkan struktur fisik penyimpanan data. Inti dari optimasi query adalah meminimal kan "jalur" pencarian untuk menemukan data yang disimpan dalam lokasi fisik.

Di dalam penggunaan SELECT * yang tidak spesifik terhadap tujuan query akan menimbulkan melambatnya pemrosesan query dibandingkan dengan penggunaan tujuan kolom. Kita perlu membatasi hasil query yang ditetapkan hanya dengan memilih kolom tertentu dari tabel, bukan semua kolom dari tabel tertentu. Hal ini menghasilkan manfaat kinerja, karena SQL Server hanya akan menampilkan kolom tertentu, tidak semua kolom pada sebuah tabel. Ini akan membantu mengurangi lalu lintas jaringan dan juga meningkatkan keseluruhan kinerja queri.

Dalam penelitian yang dilakukan juga menganjurkan untuk menghindari penggunaan klausa HAVING dalam statemen SELECT. Klausa HAVING digunakan untuk menyaring baris setelah semua baris dipilih dan digunakan seperti filter. Penggunaan HAVING Ini sangat tidak berguna dalam sebuah pernyataan SELECT. Ia bekerja dengan melewati tabel hasil akhir dari query yang mengurai baris yang tidak memenuhi kondisi HAVING [5].

Contoh:
Query dengan HAVING:
SELECT s.cust_id,count(s.cust_id)
FROM SH.sales s
GROUP BY s.cust_id
HAVING s.cust_id != '1660' AND s.cust_id != '2';

3. METODOLOGI

Metodologi yang akan digunakan adalah dengan melakukan tinjauan dan kajian dengan studi literatur yang sudah dipaparkan sebelumnya serta menggunakan analisis guna menghitung kecepatan waktu eksekusi query yang dijalankan menggunakan database institusi pendidikan.

Penggunaan query yang akan dituliskan kemudian dilakukan optimasi query sehingga mendapatkan waktu minimal dalam jumlah data yang sama yaitu:

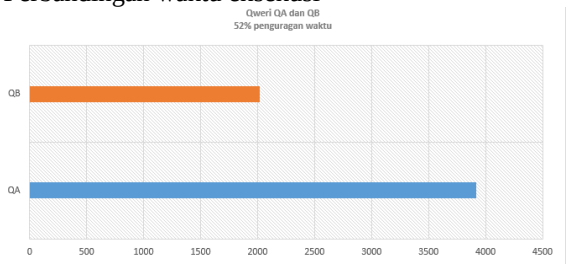
1. Penggunaan SELECT *
2. Penggunaan DISTINCT
3. Penggunaan IN
4. Penggunaan INNER JOIN dalam WHARE
5. Penggunaan EXISTS, DISTINCT
6. Penggunaan NOT IN dan LEFT JOIN

4. PEMBAHASAN

Setiap uraian pembahasan, akan dijalankan query asli dan query optimasi untuk hasil yang lebih baik dalam informasi menggunakan basis data pada SQL dengan jumlah row yang sama.

4.1. Penggunaan SELECT *

Query asli QA:
```
SELECT * FORM khs;
```
Query optimasi QB:
```
SELECT khs.nim FORM khs;
```
Perbandingan waktu eksekusi



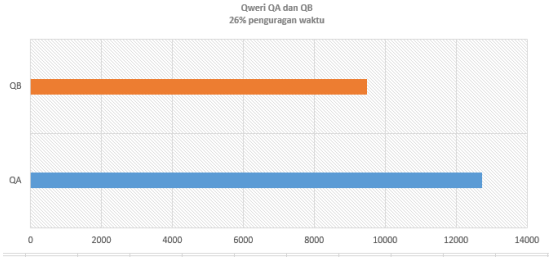
Gambar 1. *Hasil Perbedaan Waktu*.

4.2. Penggunaan DISTINCT

Query asli QA:
```
SELECT distinct *
FROM khs JOIN mhs ON khs.nim=mhs.nim
WHERE mhs.jur_id='01';
```

Query optimasi QB:
```
SELECT *
FROM khs JOIN mhs ON khs.nim=mhs.nim
WHERE mhs.jur_id='01';
```

Perbandingan waktu eksekusi



Gambar 2. *Hasil Perbedaan Waktu*.

4.3. Penggunaan IN

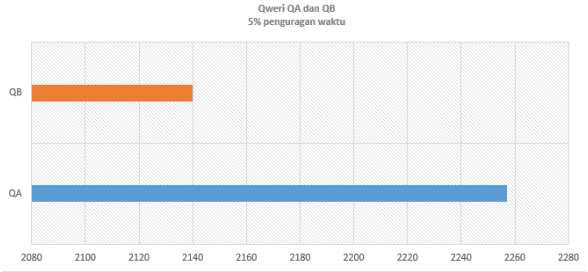
Query asli QA:

```
SELECT khs.*
FROM khs
WHERE khs.nilai_a='3' or
khs.nilai_a='4';
```

Query optimasi QB:

```
SELECT khs.*
FROM khs
WHERE khs.nilai_a in (3, 4);
```

Perbandingan waktu eksekusi



Gambar 3. *Hasil Perbedaan Waktu*.

4.4. Penggunaan INNER JOIN dalam WHERE

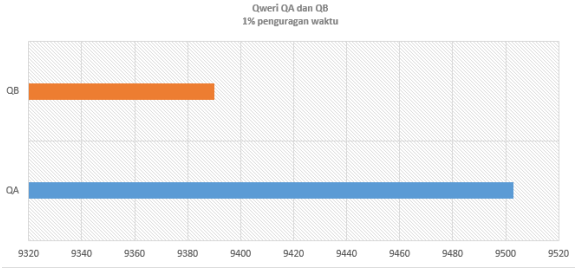
Query asli QA:

```
SELECT *
FROM mhs
INNER JOIN khs ON khs.nim=mhs.nim OR
khs.nim=mhs.nim;
```

Query optimasi QB:

```
SELECT *
FROM mhs
INNER JOIN khs ON khs.nim=mhs.nim WHERE
khs.nim=mhs.nim;
```

Perbandingan waktu eksekusi



Gambar 4. *Hasil Perbedaan Waktu*.

4.5. Penggunaan EXISTS

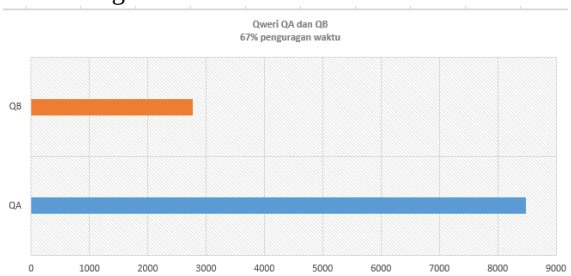
Query asli QA:

```
SELECT DISTINC *
FROM khs a, matakuliah b
WHERE (b.mk_id = a.mk_id);
```

Query optimasi QB:

```
SELECT* FROM khs a
WHERE EXISTS(SELECT 'X' FROM matakuliah
b WHERE b.mk_id = a.mk_id);
```

Perbandingan waktu eksekusi



Gambar 5. *Hasil Perbedaan Waktu*.

4.6. Penggunaan NOT IN dan LEFT JOIN

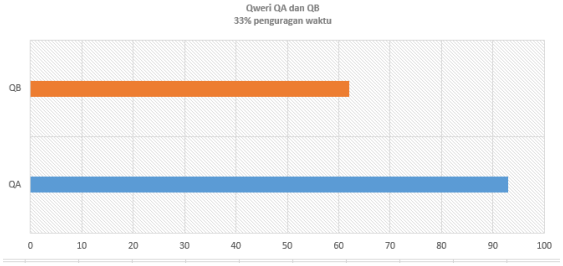
Query asli QA:

```
SELECT * FROM DOSEN
WHERE NIP NOT IN (SELECT NIP FROM
KELAS);
```

Query optimasi QB:

```
SELECT DOSEN.*
FROM DOSEN
LEFT JOIN KELAS ON DOSEN.NIP=KELAS.NIP
WHERE KELAS.NIP IS NULL;
```

Perbandingan waktu eksekusi



Gambar 6. *Hasil Perbedaan Waktu*.

5. KESIMPULAN

Berdasarkan hasil yang sudah dilakukan pembahasan diuraikan sebagai acuan dalam optimasi query yang bisa digunakan oleh DBA. sebagai berikut :

1. Dalam penggunaan SELECT * sebaiknya kita memperhatikan apa yang mau kita dapatkan pada database. Meskipun penggunaan SELECT * lebih mudah dalam penggunaanya, akan lebih menghemat waktu query apabila hanya memilih beberapa kolom yang kita tuju.

2. Dalam kasus yang sudah dijabarkan dalam pembahasan, penggunaan DISTINCT sebenarnya tidak perlu, dalam query asli tidak diperlukan DISTINCT karena table_name berisi primary key NIM, yang merupakan bagian dari kumpulan hasil.

3. Penggunaan IN dapat dilakukan untuk pengambilan indeks. Dengan menggunakan IN bisa mengurutkan sesuai daftar indeks. Dalam menerapkan ini juga harus diperhatikan bentuk data usahakan bentuk data adalah konstanta agar query bisa di eksekusi.

4. Penggunaan INNER JOIN dan WHARE menunjukan hasil yang tidak terlalu banyak perbedaan hanya selisih 1% waktu dengan jumlah data yang sama.

5. Kata kunci DISTINCT bekerja dengan memilih semua kolom dalam table. Jika kita menginginkan sub query untuk memilih semua kolom dalam tabel sebaiknya menggunakan EXISTS.

6. Penggunaan NOT IN dan LEFT JOIN mengasilkan data yang sama denga kolom dan tabel yang sudah ditentukan. Penggunaan NOT IN yag sudah dilakukan kurang begitu efektif karena membaca semua data yang ada di tabel sebelumnya, dalam implementasiya lebih baik menggunakan LEFT JOIN dengan tujuan data yang tepat.

Pengoptimalan query memiliki dampak yang baik terhadap kinerja aplikasi. Teknik optimasi terus berkembang dengan strategi pengoptimalan baru yang lebih canggih. Jadi, kita harus mencoba mengikuti cara umum seperti yang disebutkan di atas untuk mendapatkan kinerja query yang lebih baik. Optimalisasi dapat dicapai dengan beberapa upaya jika kita menjadikannya sebagai praktik umum untuk mengikuti peraturan pada optimasi query.

DAFTAR PUSTAKA

[1] Aripin. (2010). *Meningkatkan Efektifitas Pengelolaan Database Dengan Optimasi SQL*.Techno.Com, 9(1).

[2] Ema Utami dan Sukrisno. 2005. *Kosep Dasar Pengolahan dan Pemrograman Database dengan SQL server, Ms. Access, dan Ms. Visual Basic*. Yokyakarta: Penerbit Andi.

[3] Jogiyanto, HM., 1999, Analisis dan Desain Sistem Informasi, Andi Offset,Yogyakarta.

[4] Kusrini. 2007. *Strategi Perancangan dan Pengelolaan Basis Data.* Yokyakarta: Penerbit Andi.

[5] Navita Kumari, 2015. *SQL Server Query Optimization Techniques - Tips for Writing Efficient and Faster Queries.* : International Journal of Scientific and Research Publications, Volume 2, Issue 6, June 2012.